Database Systems

Lecture # 8

Dr Sherin ElGokhy

Chapter 4…to be continued: SQL

Example

EMPLOYEES

RegNo	FirstName	Surname	Dept	Salary
1201	Mahmoud	Bahaa	Administration	45
1202	Karim	Waleed	Production	36
1203	Gamal	Kamel	Administration	40
1204	Hassan	Nashaat	Distribution	45
1205	Shady	Bahaa	Planning	80
1206	Lamiaa	Shokry	Planning	73
1207	Belal	Badr	Administration	40
1208	Amira	Shaker	Production	46

The ORDER BY Clause

- Sort retrieved rows with the ORDER BY clause:
  - **ASC:** Ascending order, default
  - **DESC:** Descending order
- The ORDER BY clause must be the last clause of the SQL statement:

```
SELECT    FirstName, Surname, Salary
FROM      Employees
ORDER BY Salary ;
```

FirstName	Surname	Salary
Karim	Waleed	36
Gamal	Kamel	40
Belal	Badr	40
Mahmoud	Bahaa	45
Hassan	Nashaat	45
Amira	Shaker	46
Lamiaa	Shokry	73
Shady	Bahaa	80

Sorting

The default sort order is **ascending:**

- **Numeric** values are displayed with the lowest values first **(for example, 1 to 999).**
- **Date** values are displayed with the earliest value first **(for example,01-JAN-92 before 01-JAN-95).**
- **Character** values are displayed in the alphabetical order **(for example, “A” first and “Z” last).**
- **Null** values are displayed **last** for ascending sequences and first for descending sequences.

You can also sort by a column that is **not** in the SELECT list.

Sorting

```
SELECT    FirstName, Surname, Salary
FROM      Employees
ORDER BY Salary DESC ;
```

1

```
SELECT    FirstName, Surname, Salary*12 AnnSal
FROM      Employees
ORDER BY AnnSal ;
```

2

Examples:

1. To reverse the order in which the rows are displayed, specify the DESC keyword after the column name in the ORDER BY clause. The example sorts the result by the **highest salary.**
2. You can also **use a column alias** in the ORDER BY clause. The example sorts the data by **annual salary.**

Results of 1 and 2

(1)

FirstName	Surname	Salary
Shady	Bahaa	80
Lamiaa	Shokry	73
Amira	Shaker	46
Hassan	Nashaat	45
Mahmoud	Bahaa	45
Belal	Badr	40
Gamal	Kamel	40
Karim	Waleed	36

(2)

FirstName	Surname	AnnSal
Karim	Waleed	432
Gamal	Kamel	480
Belal	Badr	480
Mahmoud	Bahaa	540
Hassan	Nashaat	540
Amira	Shaker	552
Lamiaa	Shokry	876
Shady	Bahaa	960

Sorting

– Sorting by using the column's numeric position:

```
SELECT    FirstName, Surname, Dept, Salary
FROM      Employees
ORDER BY 3;
```

3

– Sorting by multiple columns:

```
SELECT    FirstName, Surname, Dept, Salary
FROM      Employees
ORDER BY Dept, salary  DESC;
```

4

Examples:

3. You can sort query results by specifying **the numeric position of the column in the SELECT clause.** The example sorts the result by the **Dept** name as this column is at the third position in the SELECT clause.
4. You can sort query results by **more than one column**. The sort limit is the number of columns in the given table. In the ORDER BY clause, specify the columns and separate the column names using **commas**. If you want to reverse the order of a column, specify DESC after its name.

Results of 3 and 4

(3)

FirstName	Surname	Dept	Salary
Mahmoud	Bahaa	Administration	45
Gamal	Kamel	Administration	40
Belal	Badr	Administration	40
Hassan	Nashaat	Distribution	45
Shady	Bahaa	Planning	80
Lamiaa	Shokry	Planning	73
Amira	Shaker	Production	46
Karim	Waleed	Production	36

(4)

FirstName	Surname	Dept	Salary
Mahmoud	Bahaa	Administration	45
Gamal	Kamel	Administration	40
Belal	Badr	Administration	40
Hassan	Nashaat	Distribution	45
Shady	Bahaa	Planning	80
Lamiaa	Shokry	Planning	73
Amira	Shaker	Production	46
Karim	Waleed	Production	36

Aggregate queries

- Aggregate queries **cannot** be represented in **relational algebra.**
- The result of an aggregate query depends on the consideration of sets of rows.
- SQL offers five aggregate operators:
  - **count**
  - **sum**
  - **max**
  - **min**
  - **avg**

Database for Examples

EMPLOYEES

RegNo	FirstName	Surname	Dept	Office	Salary	City
1201	Mahmoud	Bahaa	Administration	10	45	Cairo
1202	Karim	Waleed	Production	20	36	Alex
1203	Gamal	Kamel	Administration	20	40	Tanta
1204	Hassan	Nashaat	Distribution	16	45	Mansoura
1205	Shady	Bahaa	Planning	14	80	Cairo
1206	Lamiaa	Shokry	Planning	7	73	Suez
1207	Belal	Badr	Administration	75	40	Port-Said
1208	Amira	Shaker	Production	20	46	Alex

DEPARTMENT

DeptName	Address	City
Administration	Ahmed Orabi St.	Cairo
Production	Aboukeer St.	Alex
Distribution	Elkanal St.	Port-Said
Planning	Ahmed Orabi St.	Cairo
Research	Elneel St.	Asuit

Operator count

- count returns the number of rows or distinct values;
- Syntax:

count(< * | [distinct | all] AttributeList >)

- Find the number of employees:

```
SELECT count(*)
FROM   Employees;
```

Count(*)
8

Find the number of employees in the production department:

```
SELECT count(*)
FROM   Employees
Where Dept='Production' ;
```

Count(*)
2

Operator count

- **Find the number of different values on the attribute Salary for all the rows in EMPLOYEE:**

```
SELECT count(distinct Salary)
FROM   Employees;
```

Count(distinct Salary)
6

- **Find the number of rows of EMPLOYEE having a not null value on the attribute Salary:**

```
SELECT count(all Salary)
FROM   Employees;
```

Count(all Salary)
8

Sum, average, maximum and minimum

- **Syntax:**

  < sum | max | min | avg > ([distinct | all] AttributeExpr)

- **Find the sum of the salaries of the Administration department:**

```
SELECT sum(Salary) as SumSalary
FROM   Employees
WHERE Dept = 'Administration';
```

SumSalary
125

Aggregate queries with join

- **Find the maximum salary among the employees who work in a department based in Cairo:**

```
SELECT max(Salary) as MaxCairoSal
FROM   Employees, Department
WHERE Dept = DeptName and
          Department.City = 'Cairo';
```

MaxCairoSal
80

Aggregate queries and target list

- **Incorrect query:**

```
SELECT  FirstName, Surname, max(Salary)
FROM    Employees, Department
WHERE   Dept = DeptName and
        Department.City = 'Cairo';
```

Whose name? The target list must be homogeneous

Aggregate queries and target list

- **Find the maximum and minimum salaries of all employees:**

```
SELECT max(Salary) as MaxSal, min(Salary) as MinSal
FROM   Employees;
```

MaxSal	MinSal
80	36

Quiz

- **Display the name, salary and commission for all employees who earn commissions, Sort data in descending order of salary and commissions for the following schema.**

Employees (id, name, sal, comm)

Creating Groups of Data

EMPLOYEES

	DEPARTMENT_ID	SALARY	
1	10	4400	4400
2	20	13000	9500
3	20	6000	
4	50	5800	
5	50	2500	
6	50	2600	3500
7	50	3100	
8	50	3500	
9	60	4200	6400
10	60	6000	
11	60	9000	
12	80	11000	10033
13	80	10500	
14	80	8600	
...			
19	110	12000	
20	(null)	7000	

Average salary in EMPLOYEES table for each department

	DEPARTMENT_ID	AVG(SALARY)
1	10	4400
2	20	9500
3	50	3500
4	60	6400
5	80	10033.333333333333...
6	90	19333.333333333333...
7	110	10150
8	(null)	7000

Example

EMPLOYEES

RegNo	FirstName	Surname	Dept	Salary
1201	Mahmoud	Bahaa	Administration	45
1202	Karim	Waleed	Production	36
1203	Gamal	Kamel	Administration	40
1204	Hassan	Nashaat	Distribution	45
1205	Shady	Bahaa	Planning	80
1206	Lamiaa	Shokry	Planning	73
1207	Belal	Badr	Administration	40
1208	Amira	Shaker	Production	46

The GROUP BY Clause

- Queries may apply aggregate operators to subsets of rows.
- You can divide rows in a table into smaller groups by using the **GROUP BY** clause.
- All columns in the **SELECT** list that are not in group functions must be in the **GROUP BY** clause.

```
SELECT    column, group_function(column)
FROM      table
[WHERE    condition]
[GROUP BY group_by_expression]
[ORDER BY column];
```

```
SELECT    dept, MAX(salary)
FROM      employees
GROUP BY dept ;
```

The GROUP BY Clause

```
SELECT   dept, MAX(salary)
FROM     employees
GROUP BY dept ;
```

Dept	Salary
Administration	45
Administration	40
Administration	40
Planning	80
Planning	73
Production	36
Production	46
Distribution	45

Dept	MAX(Salary)
Administration	45
Planning	80
Production	46
Distribution	45

Group By Queries

- **Find the sum of salaries of all the employees of the same department:**

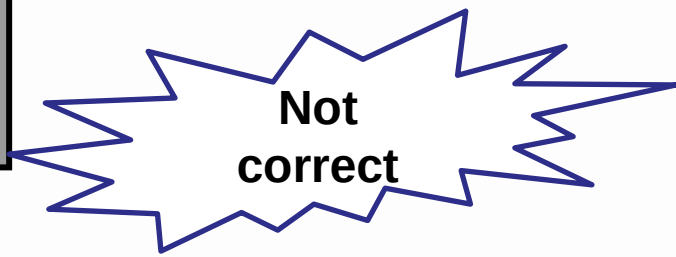
```
SELECT   Dept, sum (salary) as TotalSal
FROM     Employees
GROUP BY Dept;
```

Dept	TotalSal
Administration	125
Planning	153
Production	82
Distribution	45

Group By Queries

- **Which of the following queries is correct and which is not correct:**

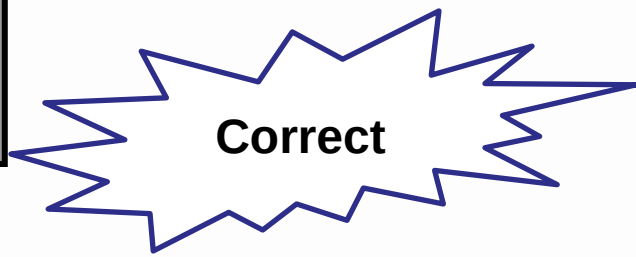
```
SELECT   Office
FROM     Employees
GROUP BY Dept ;
```



Group By Queries

- Any column or expression in the SELECT list that is not an aggregate function must be in the GROUP BY clause:

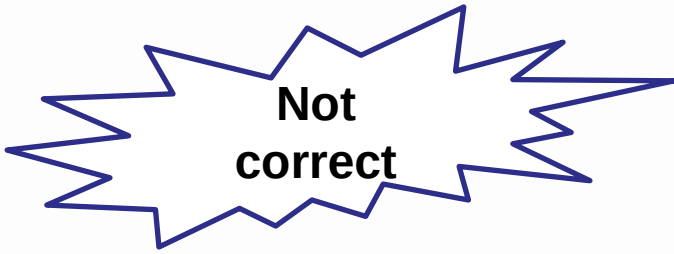
```
SELECT   Office, Dept
FROM     Employees
GROUP BY office, Dept;
```



Group By Queries

- **Which of the following queries is correct and which is not correct:**

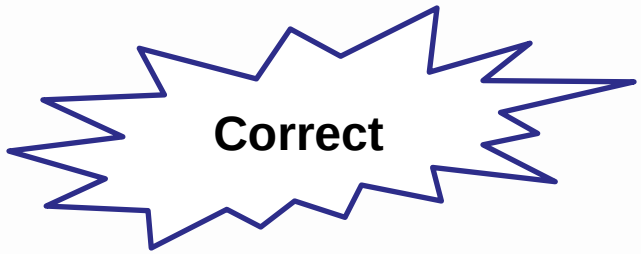
```
SELECT  DeptName, count(*), D.City
FROM  Employee E join Department D on (E.Dept = D.DeptName)
GROUP  BY  DeptName
```



Group By Queries

- Any column or expression in the SELECT list that is not an aggregate function must be in the GROUP BY clause:

```
SELECT  DeptName, count(*), D.City
FROM  Employee E join Department D on (E.Dept = D.DeptName)
GROUP  BY  DeptName, D.City
```



Illegal Queries Using Group Functions

- You cannot use the `WHERE` clause to restrict groups.
- You use the **`HAVING`** clause to restrict groups.
- You cannot use group functions in the `WHERE` clause.

Restricting Group Results with the HAVING Clause

- When you use the HAVING clause the compiler restricts groups as follows:
  1. Rows are grouped.
  2. The group function is applied.
  3. Groups matching the HAVING clause are displayed.

```
 SELECT    column, group_function
 FROM      table
[WHERE     condition]
[GROUP BY  group_by_expression]
[HAVING    group_condition]
[ORDER BY  column];
```

Group By Queries

- **Find which departments spend more than 100 on salaries:**

```
SELECT   Dept
FROM     Employees
GROUP BY Dept
HAVING   sum(salary)>100;
```

Dept
Administration
Planning

When conditions are on the result of an aggregate operator, it is necessary to use the having clause

Group By Queries

The Oracle server performs the following steps when you use the HAVING clause:

1. Rows are grouped.
2. The group function is applied to the group.
3. The groups that match the criteria in the HAVING clause are displayed.
   - The HAVING clause can precede the GROUP BY clause, but it is recommended that you place the GROUP BY clause first because it is more logical.
   - Groups are formed and group functions are calculated before the HAVING clause is applied to the groups in the SELECT list.

Group By Queries (Where or Having?)

- **Find the departments in which the average salary of employees working in office number 20 is higher than 25:**

```
SELECT   Dept
FROM     Employees
WHERE    office = '20'
GROUP BY Dept
HAVING   avg(salary)>25;
```

Only predicates containing aggregate operators should appear in the argument of the having clause.

Dept	Avg(salary)
Production	41
Administration	40

Dept
Production
Administration

Database for Examples

EMPLOYEES

RegNo	FirstName	Surname	Dept	Office	Salary	City
1201	Mahmoud	Bahaa	Administration	10	45	Cairo
1202	Karim	Waleed	Production	20	36	Alex
1203	Gamal	Karim	Administration	20	40	Tanta
1204	Hassan	Nashaat	Distribution	16	45	Mansoura
1205	Shady	Bahaa	Planning	14	80	Cairo
1206	Lamiaa	Shokry	Planning	7	73	Suez
1207	Belal	Badr	Administration	75	40	Port-Said
1208	Amira	Shaker	Production	20	46	Alex

DEPARTMENT

DeptName	Address	City
Administration	Ahmed Orabi St.	Cairo
Production	Aboukeer St.	Alex
Distribution	Elkanal St.	Port-Said
Planning	Ahmed Orabi St.	Cairo
Research	Elneel St.	Asuit

Set Queries

- **A single select cannot represent unions**
- **Syntax:**

***SelectSQL* { < union | intersect | except > [all] *SelectSQL* }**

Set Queries (Union)

- **Find the first names and surnames of the employees:**

```
SELECT FirstName as Name
FROM   Employees
UNION
SELECT Surname
FROM   Employees
```

Duplicates are removed (unless the all option is used)

Name
Mahmoud
Karim
Gamal
Hassan
Shady
Lamiaa
Belal
Amira
Bahaa
Waleed
Nashaat
Shokry
Badr
Shaker

Set Queries (Intersect)

- **Find the surnames of employees that are also first names:**

```
SELECT FirstName as Name
FROM   Employees
INTERSECT
SELECT Surname
FROM   Employees
```

Name
Karim

Equivalent to:

```
SELECT  E1.FirstName as Name
FROM    Employees E1, Employees E2
WHERE E1.FirstName = E2.Surname
```

Set Queries (Difference)

- **Find the firstnames of employees that are not also surnames:**

```
SELECT FirstName as Name
FROM   Employees
MINUS
SELECT Surname
FROM   Employees
```

```
SELECT FirstName as Name
FROM   Employees
EXCEPT
SELECT Surname
FROM   Employees
```

Name
Mahmoud
Gamal
Hassan
Shady
Lamiaa
Belal
Amira

Nested Queries

- In the where clause may appear predicates that:

– compare an attribute (or attribute expression) with the result of an SQL query; syntax:

ScalarValue Operator < any | all > SelectSQL

- any: the predicate is true if at least one row returned by SelectSQL satisfies the comparison
- all: the predicate is true if all the rows returned by SelectSQL satisfy the comparison
- We can use operators in and not in.
- The query appearing in the where clause is called nested query.

Nested Queries

- **Find the employees who work in departments in Cairo:**

```
SELECT FirstName, Surname
FROM   Employees
WHERE Dept = any ( SELECT DeptName
                   FROM   Department
                   WHERE  City= 'Cairo');
```

FirstName	Surname
Mahmoud	Bahaa
Gamal	Karim
Shady	Bahaa
Lamiaa	Shokry
Belal	Badr

Equivalent to:

```
SELECT FirstName, Surname
FROM   Employees, Department D
WHERE Dept = DeptName and
           D.City = 'Cairo'
```

Nested Queries

- **Find the departments in which there is *no one* named Bahaa:**

```
SELECT Dept
FROM   Employees
WHERE Dept <> all ( SELECT Dept
                    FROM   Employees
                    WHERE  Surname = 'Bahaa');
```

Equivalent to:

```
SELECT Dept
FROM   Employees
EXCEPT
SELECT Dept
FROM   Employees
WHERE Surname = 'Bahaa';
```

Dept
Production
Distribution

in Operator

- **Find the employees who work in departments in Cairo:**

```
SELECT FirstName, Surname
FROM   Employees
WHERE Dept = any ( SELECT DeptName
                   FROM   Department
                   WHERE  City= 'Cairo');
```

FirstName	Surname
Mahmoud	Bahaa
Gamal	Karim
Shady	Bahaa
Lamiaa	Shokry
Belal	Badr

Equivalent to:

```
SELECT FirstName, Surname
FROM   Employees
WHERE Dept in  ( SELECT DeptName
                 FROM   Department
                 WHERE  City= 'Cairo');
```

not in Operator

- **Find the departments in which there is *no one* named Bahaa:**

```
SELECT Dept
FROM   Employees
WHERE Dept <> all ( SELECT Dept
                    FROM   Employees
                    WHERE  Surname = 'Bahaa');
```

Dept
Production
Distribution

Equivalent to:

```
SELECT Dept
FROM   Employees
WHERE Dept not in ( SELECT Dept
                    FROM   Employees
                    WHERE  Surname = 'Bahaa');
```

Thanks